Toward Adaptive and Reflective Middleware for Network–Centric Combat Systems

Dr. Douglas C. Schmidt
University of California, Irvine

Dr. Richard E. Schantz
BBN Technologies

Michael W. Masters
Naval Surface Warfare Center

Dr. Joseph K. Cross
Lockheed Martin

David C. Sharp
The Boeing Company

Lou P. DiPalma
Raytheon

Software is increasingly important to the development of effective network-centric Department of Defense combat systems. Next-generation combat systems such as total ship computing environments, coordinated unmanned air vehicle systems, and national missile defense will use many geographically dispersed sensors, provide on-demand situational awareness and actuation capabilities for human operators, and respond flexibly to unanticipated run-time conditions. These combat systems will also increasingly run unobtrusively and autonomously, shielding operators from unnecessary details while communicating and responding to mission-critical information at an accelerated operational tempo. In such environments, it is hard to predict system configurations or workloads. This article describes how adaptive and reflective middleware systems (ARMS) are being developed to bridge the gap between military application programs and the underlying operating systems and communication software in order to provide reusable services whose qualities are critical to network-centric combat systems. ARMS software can adapt in response to dynamically changing conditions for the purpose of utilizing the available computer and communication resources to the highest degree possible in support of mission needs.

New and planned Department of Defense (DoD) combat systems are inherently network-centric, distributed real-time and embedded (DRE) *systems of systems*. Combat systems have historically been developed via *multiple technology bases*, where each system brings its own networks, computers, displays, software, and people to maintain and operate it. Unfortunately, not only are these *stovepipe* architectures proprietary, but by tightly coupling many functional and quality of service (QoS) aspects they impede these DRE system features:

- *Assurability* is needed to guarantee efficient, predictable, scalable, and dependable QoS from sensors to shooters.
- *Adaptability* is needed to (re)configure combat systems dynamically to support varying workloads or missions over their life cycles.
- *Affordability* is needed to reduce initial nonrecurring combat system acquisition costs and recurring upgrade and evolution costs.

In recognition of the importance of enhancing affordability, recent DoD programs such as the Aegis destroyer program [1], the New Attack Submarine program [2], the Weapons Systems Open Architecture program [3], and the Unmanned Combat Air Vehicle (UCAV) program [4] have adopted strong open systems approaches to system design and commercial-off-the-shelf (COTS) refresh strategies. Ultimately, open systems approaches are more likely to be robust with respect to change over the long life cycles typical of military systems. For example, the affordability of certain types of DoD systems such as logistics and mission planning have been improved by using COTS technologies.

However, many of today's procurement efforts aimed at integrating COTS into mission-critical DRE combat systems have largely failed to support life-cycle affordability, assurability, and adaptability effectively since they focus mainly on initial nonrecurring acquisition costs and do not reduce recurring software life-cycle costs, such as COTS refresh and subsetting combat systems for foreign military sales [5]. Likewise, many COTS products lack support for controlling key QoS properties such as predictable latency, jitter, and throughput; scalability; dependability; and security. The inability to control these QoS properties with sufficient confidence compromises combat system adaptability and assurability, e.g., a perturbation in the behavior of a COTS product that would be acceptable in commercial applications could lead to loss of life and property in military applications.

Historically, conventional COTS software has been unsuitable for use in mission-critical DRE combat systems due to either of the following:

- It is flexible and standard, but incapable of guaranteeing stringent QoS demands, which restricts assurability.
- It is partially QoS-enabled, but inflexible and non-standard, which restricts adaptability and affordability.

As a result, the rapid progress in COTS software for mainstream business information technology (IT) has not yet become as broadly applicable for mission-critical DRE combat systems. Until this problem is resolved effectively, DRE sys-

tem integrators and warfighters will not be able to take advantage of future advances in COTS software in a dependable, timely, and cost effective manner. Developing the new generation of assurable, adaptable, and affordable COTS software technologies is therefore essential for U.S. national security.

Although the near-term use of COTS software in DRE systems will be limited in scope and domain, the prospects for the longer term are much brighter. Given the proper advanced research and development (R&D) context and an effective process for transitioning R&D results, the COTS market can adapt, adopt, and implement the types of robust hardware and software capabilities needed for military applications. This process takes a good deal of time to get right and be accepted by user communities, and a good deal of patience to stay the course. When successful, however, this process results in standards that codify the best-of-breed practices and technologies and the *patterns* and *frameworks* that reify the knowledge of how to apply these practices and technologies.

Key Technical Challenges and Solutions

Today's economic and organizational constraints – along with increasingly complex requirements and competitive pressures – make it infeasible to build complex distributed real-time system software entirely from scratch. It has long been accepted that the use of commercial operating systems and communication support software is cost-effective for all but the most

1. REPORT DATE **NOV 2001**	2. REPORT TYPE	3. DATES COVERED **00-00-2001 to 00-00-2001**	
4. TITLE AND SUBTITLE **Toward Adaptive and Reflective Middleware for Network-Centric Combat Systems**		5a. CONTRACT NUMBER	
		5b. GRANT NUMBER	
		5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)		5d. PROJECT NUMBER	
		5e. TASK NUMBER	
		5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **NSWC Dahlgren Division,17320 Dahlgren Road,Dahlgren,VA,22448-5100**		8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)	
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**			
13. SUPPLEMENTARY NOTES **CROSSTALK The Journal of Defense Software Engineering, November 2001**			
14. ABSTRACT			
15. SUBJECT TERMS			

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**7**	

resource-constrained DRE systems. Increasingly, this same logic is being applied to middleware, which is reusable service/protocol component and framework software that services end-to-end and aggregate combat systems' needs [6]. Middleware bridges the gap between these areas:

- Application-level requirements and mission doctrine.
- The lower-level, underlying, localized viewpoints of the operating systems and communications support mechanisms.

From the application perspective, when middleware and the services it constitutes are combined with traditional network and operating system components, it forms the new infrastructure for developing modern network-centric combat systems. In both commercial and military systems, middleware performs functions that are essential to meeting application-level requirements. In military systems, moreover, the qualities of the services provided by the middleware are critical to the qualities of service that are presented to the end users – the warfighters.

Thus, there is a pressing need to develop, validate, and ultimately standardize a new generation of adaptive and reflective middleware systems (ARMS) technologies that will be readily available and able to support stringent combat system functionality and QoS requirements. Some of the most challenging computing and communication requirements for new and planned DoD combat systems can be characterized as follows:

- Multiple QoS properties must be satisfied in real-time.
- Different levels of service are appropriate under different configurations, environmental conditions, and costs.
- The levels of service in one dimension must be coordinated with and/or traded off against the levels of service in other dimensions to meet mission needs, e.g., the security and dependability of message transmission must be traded off against latency and predictability.
- The need for autonomous and time-critical application behavior necessitates a flexible distributed system substrate that can adapt robustly to dynamic changes in mission requirements and environmental conditions.

Adaptive middleware [3] is software whose functional and QoS-related properties can be modified in either of these ways:

- Statically, e.g., to reduce footprint, leverage capabilities that exist in specific platforms, enable functional sub-

setting, and minimize hardware and software infrastructure dependencies.
- Dynamically, e.g., to optimize system responses to changing environments or requirements, such as changing component interconnections, power-levels, CPU/network bandwidth, latency/jitter, and dependability needs.

In DRE combat systems, adaptive middleware must make these modifications dependably, i.e., while meeting stringent end-to-end QoS requirements.

Reflective middleware [7] goes a step further in providing the means for examining the capabilities it offers while the system is running, thereby enabling automated adjustment for optimizing those capabilities. Thus, reflective middleware supports more advanced adaptive behavior, i.e., the necessary adaptations can be performed autonomously based on conditions within the system, in the system's environment, or in combat system doctrine defined by operators and administrators.

Middleware Structure and Functionality

Networking protocol stacks can be decomposed into multiple layers such as the physical, data-link, network, transport, session, presentation, and application layers. Similarly, middleware can be decomposed into multiple layers such as those shown in Figure 1.
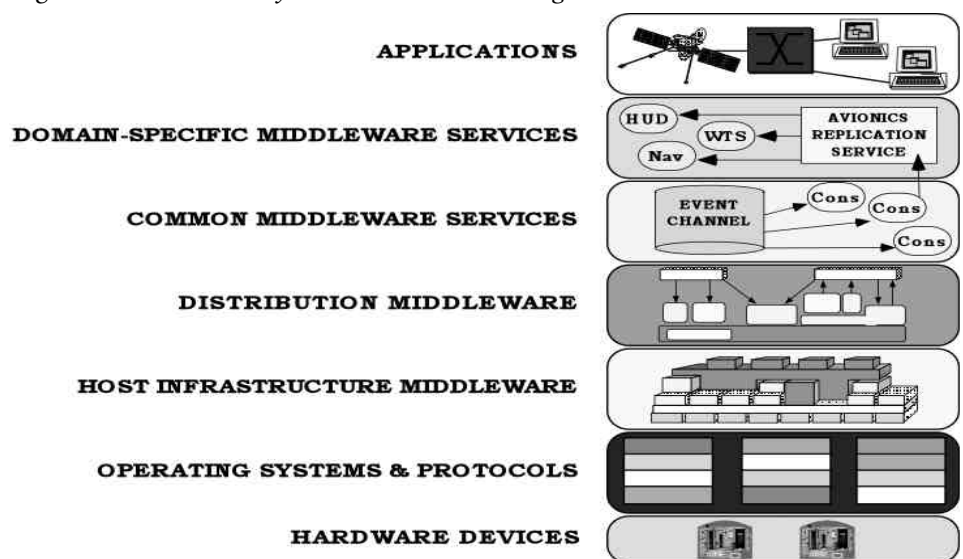
We describe each of these middleware layers and outline some of the COTS technologies in each layer that are suitable (or are becoming suitable) to meet the stringent QoS demands of DRE combat systems.

Host Infrastructure Middleware

Host infrastructure middleware encapsulates and enhances native operating system communication and concurrency mechanisms to create portable and reusable network programming components such as reactors, acceptor-connectors, monitor objects, active objects, and component configurations [8]. These components abstract away the accidental incompatibilities of individual operating systems and help eliminate many tedious, error-prone, and non-portable aspects of developing and maintaining networked applications via low-level operating system programming application program interfaces (APIs), such as Sockets or POSIX Pthreads. Examples of COTS host infrastructure middleware that are relevant for DRE combat systems include the following:

- The Adaptive Communication Environment (ACE) [9] is a portable and efficient toolkit that encapsulates native operating system network programming capabilities such as inter-process communication, static and dynamic configuration of application components, and synchronization. ACE has been used in a wide range of DoD DRE systems, including missile control, avionics mission computing, software defined radios, and radar systems.
- Real-Time Java Virtual Machines implement the Real-Time Specification for Java (RTSJ) [10]. The RTSJ is a set of extensions to Java that provide a largely platform-independent way of executing code by encapsulating the differences between real-time operating systems and CPU architectures. The key features of RTSJ deal with memory man-

Figure 1: *Middleware Layers and Their Surrounding Context*

APPLICATIONS

DOMAIN-SPECIFIC MIDDLEWARE SERVICES

COMMON MIDDLEWARE SERVICES

DISTRIBUTION MIDDLEWARE

HOST INFRASTRUCTURE MIDDLEWARE

OPERATING SYSTEMS & PROTOCOLS

HARDWARE DEVICES

agement and concurrency. Although RTSJ implementations are still in their infancy, they have generated tremendous interest in the DoD R&D and integrator communities due to their potential for reducing software development and evolution costs significantly.

Distribution Middleware

Distribution middleware defines a higher-level distributed programming model whose reusable application program interfaces and mechanisms automate and extend the native operating system network programming capabilities encapsulated by host infrastructure middleware. Distribution middleware enables developers to program distributed applications much like stand-alone applications, i.e., by invoking operations on target objects or distributed components.

At the heart of distribution middleware are QoS-enabled object request brokers, such as the Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) [4, 11]. CORBA is distribution middleware that allows objects to interoperate across networks without hard-coding dependencies on their location, programming language, operating system platform, communication protocols and interconnects, and hardware characteristics. In 1998 the OMG adopted the Real-Time CORBA specification [12], which extends CORBA with features that allow DRE applications to reserve and manage CPU, memory, and networking resources. Real-Time CORBA implementations have been used in dozens of DoD combat systems, including avionics mission computing [4], submarine combat control systems [13], signal intelligence and Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance systems, software defined radios, and radar systems.

Common Middleware Services

Common middleware services augment distribution middleware by defining higher-level, domain-independent, reusable services that have proven necessary in most distributed application contexts to deal with multi-computer environments effectively. In addition, these services provide components that allow application developers to concentrate on programming application logic, without the need to write the plumbing code needed to develop distributed applications using lower level middleware features directly.

For example, whereas distribution middleware focuses largely on managing end-system resources in support of an object-oriented distributed programming model, common middleware services focus on allocating, scheduling, and coordinating various end-to-end resources throughout a distributed system using a component programming and scripting model. Developers can reuse these services to manage global resources and perform recurring distribution tasks that would otherwise be re-implemented by each application or integrator.

Examples of common middleware services include the OMG's CORBAServices [14] and the CORBA Component Model (CCM) [15], which provide domain-independent interfaces

---

"Today's economic and organizational constraints – along with increasingly complex requirements and competitive pressures – make it infeasible to build complex distributed real-time system software entirely from scratch."

---

and distribution capabilities that can be used by many distributed applications. The OMG CORBAServices and CCM specifications define a wide variety of these services, including event notification, naming, security, and fault tolerance. Not all of these standard services are sufficiently refined today to be usable off the shelf for DRE combat systems. However, the form and content of these common middleware services will continue to mature and evolve to meet the expanding requirements of DRE.

Domain–Specific Middleware Services

Domain-specific middleware services are tailored to the requirements of particular combat system domains, such as avionics mission computing, radar processing, weapons targeting, or command and decision systems. Unlike the previous three middleware layers – which provide broadly reusable horizontal mechanisms and services – domain-specific middleware

services are targeted at vertical market segments. From a COTS perspective, domain-specific services are the least mature of the middleware layers today. This immaturity is due in part to the historical lack of distribution middleware and common middleware service standards, which are needed to provide a stable base upon which to create domain-specific middleware services. Since they embody knowledge of a domain, however, domain-specific middleware services have the most potential to increase the quality and decrease the cycle time and effort that DoD integrators require to develop particular classes of DRE combat systems.

A mature example of domain-specific middleware services appears in the Boeing Bold Stroke architecture [4]. Bold Stroke uses COTS hardware and middleware to produce a non-proprietary, standards-based component architecture for military avionics mission computing capabilities, such as navigation, data link management, and weapons control. A driving objective of Bold Stroke was to support reusable product-line applications, leading to a highly configurable application component model and supporting middleware services. The domain-specific middleware services in Bold Stroke are layered upon common middleware services (the CORBA Event Service), distribution middleware (Real-Time CORBA and the tactical air operations object request broker [16]), and infrastructure middleware advanced computing environment, and have been demonstrated to be highly portable for different COTS operating systems (e.g., VxWorks), interconnects (e.g., VME), and processors (e.g., PowerPC).

Recent Progress

Significant progress has occurred during the last five years in DRE middleware research, development, and deployment within the DoD, stemming in large part from the following trends:

Maturation of Standards

During the past decade, middleware standards have been established and have matured considerably with respect to DRE requirements. For example, the OMG has recently adopted the following DRE-related specifications:

- Minimum CORBA removes non-essential features from the full OMG CORBA specification to reduce footprint so that CORBA can be used in memory-constrained embedded systems.

- Real-Time CORBA includes features that allow applications to reserve and manage network, CPU, and memory resources predictably end to end.
- CORBA Messaging exports additional QoS policies such as timeouts, request priorities, and queuing disciplines to applications.
- Fault-Tolerant CORBA uses entity redundancy of objects to support replication, fault detection, and failure recovery.

Robust and interoperable implementations of these CORBA capabilities and services are now available from multiple vendors. Moreover, emerging standards such as Dynamic Scheduling Real-Time CORBA, Real-Time CORBA publish-subscribe services, the Real-Time Specification for Java, and the Distributed Real-Time Specification for Java are extending the scope of open standards for a wider range of DoD applications.

Dissemination of Patterns, Frameworks

A substantial amount of R&D effort during the past decade has also focused on the following means of promoting the development and reuse of high quality middleware technology:

- Patterns codify design expertise that provides time-proven solutions to commonly occurring software problems that arise in particular contexts [17]. Patterns can simplify the design, construction, and performance tuning of DRE applications by codifying the accumulated expertise of developers, architects, and systems engineers who have already confronted similar problems successfully.
- Frameworks are concrete realizations of related patterns [18] that provide an integrated set of components that collaborate to provide a reusable architecture for a family of related applications. Middleware frameworks include strategized selection and optimization patterns so that multiple, independently developed capabilities can be integrated and configured automatically to meet the functional and QoS requirements of particular DRE applications.

Historically, the knowledge required to develop predictable, scalable, efficient, and dependable mission-critical DoD DRE combat systems has existed largely in programming folklore, the heads of experienced researchers and developers, or buried deep within millions of lines of complex source code. Moreover, docu-

menting complex systems with today's popular software modeling methods and tools, such as the Unified Modeling Language (UML), only capture how a system is designed, but do not necessarily articulate why a system is designed in a particular way, which complicates subsequent software evolution and optimization.

Middleware patterns and frameworks help address these problems by systematically capturing combat system design expertise in a readily accessible and reusable format, thereby raising the level at which systems engineers and application developers approach the decision making and implementation of their systems. Two efforts to provide suitable guid-

---

"Given the proper advanced R&D context and an effective process for transitioning R&D results, the COTS market can adapt, adopt, and implement the types of ... capabilities needed for military applications ..."

---

ance for the development of military systems are the New Attack Submarine (NAS) [2] and the Aegis Shipbuilding Program. NAS developed a guidance document detailing allowable standards for the NAS C3I system, and the Aegis program developed a guidance document for Baseline 7 phase I [19]. These documents were instrumental in guiding the design of these systems.

Much of the pioneering R&D on middleware patterns, frameworks, and standards for DRE combat systems has been conducted as part of the Defense Advanced Research Projects Agency's (DARPA) Information Technology Office Quorum Program [20], which played a leading role in the following:

- Demonstrating the viability of host infrastructure middleware and distribution middleware for DoD combat systems by providing the foundation for managing key QoS attributes such as real time behavior, dependability, and system survivability from a network-centric middleware perspective.
- Transitioning a number of new mid-

dleware perspectives and capabilities into DoD acquisition programs [4, 21] and commercially supported products.
- Establishing the technical viability of collections of systems that can dynamically adapt [3] their collective behavior to varying operating conditions, in service of delivering the appropriate application level response under these different conditions.

The Quorum program focused heavily on CORBA open systems middleware and yielded many results that transitioned into standardized service definitions and implementations for the Real-Time [4] and Fault-Tolerant [22] CORBA specification and production. Quorum is an example of how a focused government R&D effort can leverage its results by exporting them into, and combining them with, other on-going public and private activities by using a common open middleware substrate. Prior to the viability of standards-based COTS middleware platforms, these same R&D results would have been buried within custom or proprietary systems, serving only as an existence proof, rather than as the basis for realigning the DoD R&D and integrator communities.

Successful DoD technology transition most often results from a partnership between technology developers and technology users. One of the most successful examples of such partnerships is the joint DARPA/Aegis High Performance Distributed Computing program (HiPer-D). Through the use of prototyping and system-scale experiments, this program has demonstrated the effectiveness of a number of DARPA and standards-based COTS technologies for building DRE combat systems that are efficient, scalable, fault tolerant, and flexible in their design and operation.

Looking Ahead

Due to advances in COTS technologies outlined earlier, host infrastructure middleware and distribution middleware have now been demonstrated and deployed in a number of mission-critical DRE combat systems. Since off-the-shelf middleware technology has not yet matured to cover the realm of large-scale dynamically changing systems, however, COTS DRE middleware has been applied to relatively small-scale and statically configured embedded systems. To satisfy the highly application- and mission-specific QoS requirements in network-centric *system-to-system environments*, DRE middleware
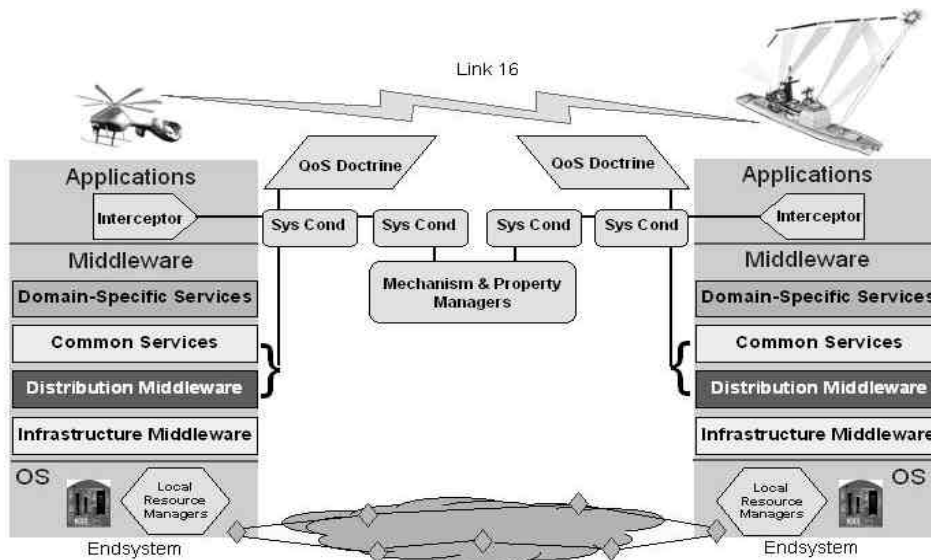
Figure 2: *Decoupling Functional and QoS Attribute Paths in QuO*

must therefore be enhanced to support common and domain-specific middleware services that can manage the following resources effectively:

- Communication bandwidth, e.g., network level status information and management services, scalability to 102 subnets and 103 nodes, and dynamic connections with controlled and reserved bandwidth to enhance real-time predictability.
- Distributed real-time scheduling and allocation of DRE system artifacts (such as CPUs, networks, UAVs, missiles, torpedoes, radar, illuminators, etc), e.g., fast and predictable behavior of widely dispersed components that use the managed communication capabilities and bandwidth reservations.
- Distributed system dependability, e.g., policy-based selection of replication options to control footprint and reactive behavior to failures.
- Distributed system security, e.g., dynamically variable object access control policies and effective, combined real-time dependability, and security interactions.

Ironically, there is little or no scientific underpinning for QoS-enabled resource management, despite the demand for it in most distributed systems [23]. Today's system designers develop concrete plans for creating global, end-to-end functionality. These plans contain high-level abstractions and doctrine associated with resource management algorithms, relationships between these, and operations upon these. There are few techniques and tools, however that enable users, i.e., commanders, administrators, and operators, and developers, i.e., systems engineers and application designers, and/or applications to express such plans systematically, reason about and

refine them, and have these plans enforced automatically to manage resources at multiple levels in network-centric combat systems.

To address this problem, the R&D community needs to discover and set the technical approach that can significantly improve the effective utilization of networks and end-systems that DRE combat systems depend upon by creating middleware and distributed resource management technologies and tools that can automatically allocate, schedule, control, and optimize customizable – yet standards-compliant and verifiably correct – software-intensive systems. To promote a *common technology* base, the interfaces and (where appropriate) the protocols used by the middleware should be based on established or emerging industry or DoD standards that are relevant for DRE combat systems. However, the protocol and service *implementations* should be customizable – statically and dynamically – for specific DoD DRE combat system requirements.

To achieve these goals, middleware technologies and tools need to be based upon some type of layered architecture along with QoS adaptive middleware services such as the one shown in Figure 2 and based on empirical investigations of this type of capability [3].

The Quality Objects (QuO) [24] project is an example of such a layered architecture designed to manage and package adaptive QoS capabilities as common middleware services. The QuO architecture decouples DRE middleware and applications along the following two dimensions:

- Functional paths are flows of information between client and remote server applications. In distributed systems, middleware ensures that this information is exchanged efficiently, pre-

dictably, dependably, and securely between remote peers, and in a manner that scales well to large configurations. The information itself is largely application-specific and determined by the functionality being provided (hence the term *functional path*).
- QoS attribute paths are responsible for determining how well the functional interactions behave end to end with respect to key DRE system QoS properties such as the following:
  1. How and when resources are committed to client/server interactions at multiple levels of distributed systems.
  2. The proper application and system behavior if available resources are less than the expected resources.
  3. The failure detection and recovery strategies necessary to meet end-to-end dependability requirements under anomalous conditions.

In next-generation combat systems, the middleware – rather than operating systems or networks in isolation – will be responsible for separating DRE system QoS attribute properties from the functional application properties. Middleware will also coordinate the QoS of various DRE system and application resources end to end. The architecture in Figure 2 enables these properties and resources to change independently, e.g., over different distributed system configurations for the same application.

The architecture in Figure 2 is based on the expectation that QoS attribute paths will be developed, configured, monitored, managed, and controlled by a different set of specialists (such as systems engineers, administrators, operators, and perhaps someday automated agents) and tools than those customarily responsible for programming functional paths in DRE systems. The middleware is therefore responsible for collecting, organizing, and disseminating QoS-related meta-information that is needed to do the following:

- Monitor and manage how well the functional interactions occur at multiple levels of DRE systems.
- Enable the adaptive and reflective decision-making needed to support QoS attribute properties robustly in the face of rapidly changing mission requirements and environmental conditions.

Researching and developing these middleware capabilities is crucial to ensure that the aggregate behavior of future network-centric combat systems is dependable, despite local failures, transient overloads, and dynamic functional or QoS reconfigurations.

To simultaneously enhance assurability,

adaptability, and affordability, the middleware techniques and tools developed in future R&D programs increasingly need to be application-independent, yet customizable within the interfaces specified by a range of open standards such as these:

- The OMG Real-Time CORBA specifications and The Open Group's QoS Task Force.
- The Java Expert Group Real-Time Specification for Java (RTSJ) and the emerging Distributed RTSJ.
- The IEEE Real-Time Portable Operating System (POSIX) specification.

Conclusions

As a result of much previous R&D and transition experience, network-centric systems today are constructed as a series of layers of intertwined technical capabilities and innovations. The main emphasis at the lower layers is in providing the core computing and communication resources and services that drive network-centric computing: the individual computers, the networks, and the operating systems that control the individual host and the message level communication.

At the upper layers, various types of middleware are starting to bridge the previously formidable gap between the lower-level resources and services and the abstractions that are needed to program, organize, and control systems composed of coordinated, rather than isolated, components. Key capabilities in the upper layers include common and domain-specific middleware services that provide the following:

- Enforcing real-time behavior across computational nodes.
- Managing redundancy across elements to support dependable computing.
- Controlling end-to-end adaptive behavior in responding to changes in operating conditions while continuing to meet application needs.

These new middleware services make the coordinated use of multiple computing elements feasible and affordable by controlling the hardware, network, and end-system mechanisms that affect mission, system, and application QoS delivery and tradeoffs that are needed to deliver the right QoS at the right time under the prevailing conditions.

Adaptive and reflective middleware systems (ARMS) is a key emerging paradigm that will help to simplify the development, optimization, validation, and integration of DRE middleware in DoD combat systems. In particular, ARMS will allow researchers and system integrators to develop and evolve complex combat systems assurably, adaptively, and affordably through the following:

- Devising optimizers, meta-programming techniques, and multi-level distributed dynamic resource management protocols and services for ARMS that will enable DoD DRE systems to configure standard COTS interfaces without the penalties incurred by today's conventional COTS software product implementations. Many network-centric DoD combat systems require these DRE middleware capabilities.
- Standardizing COTS at the middleware level, rather than just at lower hardware/networks/operating system levels. The primary economic benefits of middleware stem from extending standardization up several levels of abstraction so that DRE middleware technology is readily available for COTS acquisition and customization.

As COTS implementations of middleware standards mature in their functional quality and QoS, they are helping to lower the total ownership costs of combat systems. For example, Real-Time and Fault-Tolerant CORBA implementations are creating a common base of COTS technology that enables complex DRE middleware capabilities to be reconfigured and reused, rather than reinvented repeatedly or reworked from proprietary stovepipe architectures that are inflexible and expensive to maintain, evolve, and optimize. Additional information on middleware for DRE systems is available at <www.ece.uci.edu/~schmidt/TAO.html>.◆

References

1. Holzer, R. "U.S. Navy Looking for More Adaptable Aegis Radar," Defense News 18 Sept. 2000.
2. New Attack Submarine Open System Implementation, Specification and Guidance, Aug. 1994.
3. Loyall J. L., et al. "Comparing and Contrasting Adaptive Middleware Support in Wide-Area and Embedded Distributed Object Applications." Proceedings of the 21st IEEE International Conference on Distributed Computing Systems. Phoenix, AR. 16-19 Apr. 2001.
4. Sharp, David C. "Reducing Avionics Software Cost Through Component Based Product Line Development," Software Technology Conference. Salt Lake City, Apr. 1998.
5. Clapp, J., and A. Taub, eds. A Management Guide to Software Maintenance in COTS-Based Systems MP 98B0000069. Bedford, MA: The MITRE Corporation, Nov. 1998.
6. Schantz, R., and D. Schmidt. "Middleware for Distributed Systems: Evolving the Common Structure for Network-Centric Applications." Encyclopedia of Software Engineering. Wiley & Sons, 2001.
7. Blair, G. S., F. Costa, G. Coulson, and H. Duran, et al. "The Design of a Resource-Aware Reflective Middleware Architecture." Proceedings of the 2nd International Conference on Meta-Level Architectures and Reflection. St.-Malo, France: Springer-Verlag, LNCS, Vol. 1616, 1999.
8. Schmidt D., M. Stal, H. Rohnert, and F. Buschmann F., eds. Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects. Wiley and Sons, 2000.
9. Schmidt D., and S. Huston, eds. C++ Network Programming: Resolving Complexity with ACE and Patterns. Reading, MA: Addison-Wesley, 2002.
10. Bollella, G., and J. Gosling. "The Real-Time Specification for Java." Computer June 2000.
11. Object Management Group, The Common Object Request Broker: Architecture and Specification Rev. 2.4. OMG Technical Document formal/00-11-07, Oct. 2000.
12. Schmidt, D., and F. Kuhns, eds. "An Overview of the Real-Time CORBA Specification." IEEE Computer Magazine June 2000.
13. DiPalma, L., "The Infusion of CORBA into the U.S. Navy's Submarine Fleet," Software Technology Conference. Salt Lake City, May 1999.
14. Object Management Group. CORBA-Services: Common Object Service Specification. OMG Technical Document Formal/98-12-31.
15. Object Management Group, CORBA Component Model Joint Revised Submission. OMG Document orbos/99-07-01.
16. Schmidt D., Levine D., and Mungee S., eds. "The Design and Performance of the TAO Real-Time Object Request Broker," Computer Communications Special Issue on Building Quality of Service into Distributed Systems 21.4 (1998).
17. Gamma E., R. Helm, R. Johnson, J. Vlissides, eds. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.
18. Johnson R., "Frameworks = Patterns + Components." Communications of the ACM 40.10, Oct. 1997.
19. Guidance Document for Aegis Baseline 7 Phase 1 and II Specification Development: Information Architecture and Baseline Applicability, Ver. 1.0, 20 Mar. 1998.
20. The Quorum Program, Defense Ad-

vanced Research Projects Agency, <www.darpa.mil/ito/research/quorum/index.html> 1999.

21. Guidance Document for Aegis Open Architecture Baseline Specification Development, Ver. 2.0 (Draft), 5 July 2001.

22. Object Management Group. Fault Tolerance CORBA Using Entity Redundancy RFP. OMG Document orbos/98-04-01 edition, 1998.

23. Narain S., R. Vaidyanathan, S. Moyer, W. Stephens, K. Parameswaran, and A. Shareef, eds. "Middleware For Building Adaptive Systems via Configuration," Workshop. ACM Optimization of Middleware and Distributed Systems (OM 2001), Snowbird, UT. June 2001.

24. Zinky, J.A., D.E. Bakken, and R.E. Schantz, eds. "Architectural Support for Quality of Service for CORBA Objects." Theory and Practice of Object Systems 3.1, Apr. 1997.

About the Authors

Douglas C. Schmidt, Ph.D., is an associate professor in the Electrical and Computer Engineering Department at the University of California, Irvine. He currently serves as deputy director of the Defense Advanced Research Projects Agency's Information Technology Office, where he leads the national effort on distributed object computing middleware research and development. His research focuses on design patterns, implementation, and experimental analysis of object-oriented frameworks that facilitate the development of high-performance, real-time distributed object computing systems on parallel processing platforms running over high-speed networks and embedded system interconnects.

Electrical and Computer
Engineering Department
University of California, Irvine
Irvine, CA 92697-2625
Phone: (949) 824-1901
Fax: (949) 824-2321
E-mail: schmidt@uci.edu

Richard E. Schantz, Ph. D., is a principal scientist at BBN Technologies in Cambridge, Mass. His research has been instrumental in defining and evolving the concepts underlying middleware since its emergence in the early days of the Internet. He was directly responsible for developing the first operational distributed object computing capability and transitioning it to production use. More recently, he has led research efforts toward developing and demonstrating the effectiveness of middleware support for adaptively managed Quality Of Service control, as principal investigator on a number of key Defense Advanced Research Projects Agency, Information Technology Office projects.

BBN Technologies
10 Moulton Street
Cambridge, MA 02138
Phone: (617) 873-3550
Fax: (617) 873-4328
E-mail: schantz@bbn.com

Michael W. Masters serves as chief scientist for the U.S. Navy's High Performance Distributed Computing program (HiPer-D), a joint effort between the Aegis shipbuilding program and several Defense Advanced Research Projects Agency, Information Technology Office programs. HiPer-D is defining a new distributed real-time computing architecture for shipboard use. Masters is co-inventor of a technology called dynamic resource management, an enterprise-wide system control capability that allows large-scale real-time systems to dynamically reconfigure themselves to adapt to varying environments, changing mission demands and current resource availability.

NSWC Dahlgren Division
17320 Dahlgren Road,
Dahlgren, VA 22448-5100
Phone: (540) 653-1611
Fax: (540) 653-6415
E-mail: mastersmw@nswc.navy.mil

Joseph K. Cross, Ph.D., is a senior staff system engineer at Lockheed Martin Tactical Systems, Eagan, Minn. He is currently serving as principal investigator of the Meta-Interfaces for Embedded Real-Time Systems project in Defense Advanced Research Projects Agency, Information Technology Office. His other activities focus on middleware for Navy standard products and mechanisms for automatic configuration of complex communication systems.

Lockheed Martin Tactical Systems
P.O. Box 64525 MS U2N27
St. Paul, MN 55164-0525
Phone: (651) 456-7316
Fax: (651) 456-2078
E-mail: joseph.k.cross@lmco.com

David C. Sharp is a Technical Fellow at Boeing Phantom Works in St. Louis, Mo. As lead architect and core architecture team leader for Boeing's Bold Stroke product line avionics software initiative, Sharp spearheaded the development, documentation, and presentation of the Bold Stroke Software Architecture. This reusable product-line software architecture is used as the basis for avionics program work on a range of Boeing production and experimental aircraft programs, and as the foundation for several U.S. government-sponsored research and development programs. Sharp serves as principal investigator for a number of Defense Advanced Research Projects Agency, Information Technology Office and Air Force Research Laboratory programs.

The Boeing Company
P.O. Box 516
St. Louis, MO 63166
Phone: (314) 233-5628
Fax: (314) 233-8323
E-mail: david.sharp@boeing.com

Lou P. DiPalma is the manager of the Sub-Surface Warfighter Information Center Systems Engineering Department of the Portsmouth, R.I. headquarters of the Naval & Maritime Integrated Systems Operation of the Raytheon Electronic Systems Company. DiPalma has been involved in the design and development of Submarine Combat Control Systems including the New Attack Submarine CC, the Combat Control System Mk 2 and AN/BSG-1 Weapon Launching System Programs. He has been actively involved with the infusion of new technology into the aforementioned systems, including Common Object Request Broker Architecture (CORBA) and Real-Time CORBA.

Raytheon Electrical Systems
Naval and Maritime Integrated Systems
1847 West Main Road
Portsmouth, RI 02871
Phone: (401) 842-5592
Fax: (401) 842-5232
E-mail: louis_p_dipalma@raytheon.com